Research Article

Safety Functions in Software Development Life Cycle

G. SATYANARAYANA
Associate Professor & HOD,
Department of Information Technology,
Dadi Institute of Engineering & Technology,
NH – 5, Anakapalle

Abstract:- **Control design and real-time implementation are usually performed in isolation. The effects of the computer implementation on control system performance are still evaluated on the last phases of the development cycle. It is expected that modeling the computer implementation in order to simulate its impact on control would help reducing the length and the effort of the development cycle. This paper proposes ideas towards achieving these objectives. Today the growth of software development has increased vastly and the software development industry culminated the apex of all the other industries. This is because of a factor that the software industry became basis for the growth and progression of all the other industries of the world. As a result the burden for a software developer has increased to some extent. So there is a need for us to take care and provide more and more security for the Software application we develop. So here we have provided some safety precautions which must be taken while undergoing the process of SDLC.**

Key Words: **Analysis, Design, Software Development Life Cycle, Software Engineering, System Safety**

I. INTRODUCTION

A Safety Critical System is a system where human safety is dependent upon the correct operation of system. Safety is considered not only for software elements but also for hardware, electrical hardware, operators or users etc. If the failure of a system could lead to consequences that are determined to be unacceptable then the system is safety critical.

Safety-critical systems, a term whose customary meaning is systems whose failure might danger human life, lead to substantial economic loss, or cause extensive environmental damage. Many modern systems depend on computers for their correct operation. The future is likely to increase dramatically the number of computer systems that we consider to be safety-critical. The dropping cost of hardware, the improvement in hardware quality, and other technological developments ensure that new applications will be sought in many domains.

If software engineering is to make solid progress towards becoming a mature discipline, then it must move in the direction of establishing standardized, disciplined methods and processes that can be used systematically by practitioners in carrying out their routine software development tasks. We note that such standardized methods and processes should not be totally inflexible, but indeed must be tailor-able and evolvable to enable different practitioners to respond to what is known to be a very wide range of software development situations in correspondingly different ways, and to enable them to change and adapt their processes as these situations change. Regardless of this, however, the basis of a mature discipline of software engineering seems to us to entail being able to systematically execute a clearly defined process in carrying out these tasks. In this paper we refer to a process as being "systematic" if it provides precise and specific guidance for software practitioners to rationally carry out the routine parts of their work. As design is perhaps the most crucial task in software development, it seems particularly important that software design processes be clearly defined in such a way as to be more systematic.

Failures in safety critical systems may result in death of people, pollution of the environment or destruction of property and equipment. In industrial automation, highly dependable or fail-safe systems are in use to reduce the risk of harms. In case of failure, fail-safe systems are required to fail in a way not to harm people, environment or property. The systems have the well defined task to achieve and maintain a safe state in case of a fault.

II. SOFTWARE SYSTEM SAFETY

Software safety is best addressed by system safety engineers. Software safety must be approached from a system level. A boundary on software safety engineering tasks should be drawn so that safety analysis does not extend into the software quality tasks, such as the auditing of development, maintenance, and configuration control procedures. System safety is to identify and assess risks and assist in minimizing the hazards.

Research Article

III. SOFTWARE DEVELOPMENT LIFE CYCLE

Software Development Life Cycle or SDLC is a model of a detailed plan on how to create, develop, implement and eventually fold the software. It's a complete plan outlining how the software will be born, raised and eventually be retired from its function. Although some of the models don't explicitly say how the program will be folded, it's already common knowledge that software will eventually have it's ending in a never ending world of change web, software and programming technology.

SDLC or the Software Development Life Cycle refers to the steps involved in creating software from scratch. During earlier days, software used to be scarce and its creation was comparatively easier. Developers could create any programs they want and experience no competition. In present times, software released to the public can easily be compared to other software. Software can even be considered as a version of another program without any credit. That's why careful planning has to be made before a single code is written. Analysis has to be done before anything else is made. SDLC is known as the traditional way of executing and implementing a program.

The following are the basic stages of SDLC:

3.1. Market Research

Before anything is implemented or written, SDLC should always understand their customers in the first place. It's just basic business to consider what is required and what's in demand in today's market. An abstract program could be created from this stage. Since researchers are technically not the developers, all they could do is create a list of what could be used by the public. This will basically be the mission of the software – to meet and exceed the needs that are laid out by the market research team.

3.2. Hardware Preparation

Once the need has been set, it is time to determine the required hardware to create the specific software. Hardware usually comes after research since the company does not know the actual requirement. They still need a plan before hardware could be purchased or upgraded.

3.3. Software Analysis

Once the hardware is in place, it is time to go to the specifics of the software. Developers will have to work closely with the research team and present a specific solution for their needs. In this stage of software development, the framework of the software is created. The software gains some foundation as the developers will know what to work on.

3.4. Software Design

The specifics of the software are elaborated in this stage of web development. Developers will now create a layout of how the software should perform. The workflow is eventually established and the front-end of the software is laid out. The rough design of the software is also presented. Analysis is constantly made on the look of the layout since it will be the first thing that customers will see. This stage is very crucial since the foundation is elaborated and things are extended for the first time. The end result and structure will be based on the initial design. The workflow is laid out and it will be the backbone of the coding structure. Developers have to ensure that each step has been explained carefully if not other developers will find it difficult to understand the software.

3.5. Coding

If you compare it to a business model, coding is the "operations" of the software development plan. Everything is realized during this stage. The colors and the functions are gradually developed at this point. Developers will make use of different kinds of coding techniques in their preferred platforms. They may create different platforms or create a mash-up of different platforms to different programs. Whatever their coding technique is, they will be adhering to the plans set out by the workflow. Once the program is finished, a prototype is set out for the next stage of the process.

3.6. Testing

Of course the program, on its first run will not perfect or wouldn't exactly work as planned. It has to go through rigorous testing. In this stage, bugs are found and some irregularities in the software are somehow fixed. If something goes wrong with the program, it could be fixed with a simple change in codes but if the program is not working as planned, it will be returned to the developers for another round of coding. Testing will make sure the customer gets what they want all the time. Nothing is compromised during this stage. Depending on the program's coding efficiency, testing could last for a very long time.

3.7. Implementation

Once the testing of program has been completed the software is ready for implementation. For various developers, implementation could mean releasing the software for public beta-testing. That means everyone can use this program but the service is not yet complete. This implementation is used to give them a glimpse of something better. The beta release

is also used to test the program if it can withstand thousands of users at the same time. Of course, in theory it has to work but there is software that, for some reason crashes when it reaches a certain number of users at the same time. These are usually done for gaming programs. Once the program has been under beta version for some time without any glitch, the final version is released to the public.

3.8. Closing

This is not very popular with the US developers, but in the UK, closing refers to the final version of the software. Everything is already in place once closing has been posted – the documentation, source codes and business plans were also implemented at the same time.

3.9. Maintenance

Once the program has been released, it will not be left out alone. Developers will still work on the software full time as they will monitor its popularity and performance. No one knows if something will happen to a program. This could be the easiest part of the software development stage or the most difficult – depending on the programs efficiency. That is why even at an early stage, coding and design has to be tight and precise.

IV. TERMINOLOGY

Process – The IEEE defines a process as "a sequence of steps performed for a given purpose". A secure software process can be defined as the set of activities performed to develop, maintain, and deliver a secure software solution. Activities may not necessarily be sequential; they could be concurrent or iterative.

Process model – A process model provides a reference set of best practices that can be used for both process improvement and process assessment. Process models do not define processes; rather, they define the characteristics of processes. Process models usually have architecture or a structure. Groups of best practices that lead to achieving common goals are grouped into process areas, and similar process areas may further be grouped into categories. Most process models also have a capability or maturity dimension, which can be used for assessment and evaluation purposes.

Standards – Standards are established by some authority, custom, or by general consent as examples of best practices. Standards provide material suitable for the definition of processes.

Assessments, Evaluations, Appraisals – All three of these terms imply comparison of a process being practiced to a reference process model or standard. Assessments, evaluations, and appraisals are used to understand process capability in order to improve processes. They help determine whether the processes being practiced are adequately specified, designed, integrated, and implemented to support the needs, including the security needs, of the software product. They are also important mechanisms for selecting suppliers and then monitoring supplier performance.

Software Assurance – SA is defined as "the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its life cycle, and that the software functions in the intended manner". In the Capability Maturity Model for Software, the purpose of "software assurance" is described as providing appropriate visibility into the process being used by the software projects and into the products being built.

V. A NEW APPROACH TO PROVIDE SECURITY FOR SDLC PROCESS

Till now we have seen various things like the safety systems, real world situations, SDLC process and all the main divisions of it. But here the question arises - how to provide safety in each & every phase of the development life cycle? The analysis phase is the main step which everyone takes before performing a task. So, one should be good at analyzing to get the successful outcome of the project according to user requirements.

So now we are ready with the idea regarding an application development. Now we have to check the latest trend of the application which can only done by the Market research phase.

Every development processes the main entities:

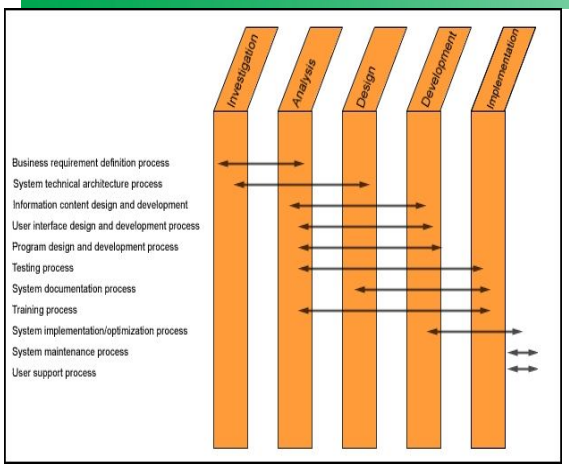
- ➔ Hardware
- ➔ Software

We should prepare enough hardware devices according to the requirement so that the application or the software can be developed properly without any flaws. While doing these processes we need to keep one thing in our mind that is "Final code preparation".

So for the preparation of a software application we must have an operating system to be loaded in the hardware component and also take the help of a software component for producing an application aspired by the end-user.

Now comes to System design. At design time, an increasingly important time is dedicated to the writing of the software that will be executed on the computer architecture.

Research Article



The primary objective of the design phase is to create a design that satisfies the agreed application requirements. In the design phase the SDLC process continues to move from the "what" questions of the analysis phase to the "how" questions.

The requirements prototype that was developed earlier during the analysis phase is gradually improved and extended to include all the specified functions of the application.

Also, the planning of the system documentation process should be started.

The design phase mainly deals with the modeling issues. As we know that we need to generate a plan of the working and entire design of a system by using Unified Modeling Language. Unified Modeling Language is the one of the most exciting tools in the world of system development today. Because UML enables system builders to create blue prints that capture their visions in a standard, easy to understand way and communicate them to others.

VI. REFERENCES

[1] Robyn R. Lutz, "Software Engineering for Safety: a Roadmap", *Proceedings of the Conference on The Future of Software Engineering*, June 04-11, 2000, Limerick, Ireland, pp. 213-226

[2] Alan C. Tribble et al. "Software Safety Analysis of a Flight Guidance System", *Proceedings of the 21st Digital Avionics Systems Conference* (DASC'02), Irvine, California, Oct. 27-31, 2002

[3] Debra S. Herman, "*Software Safety and Reliability Basics*", (ch.2), Software Safety and Reliability: Techniques, Approaches, and Standards of Key Industrial Sectors Wiley-IEEE Computer Society Press, 2000

[4] Dale M. Gray. *Frontier Status Report* #203, 19 May 2000, www.asi.org

[5] John C. Knight. "Safety Critical Systems: Challenges and Directions" *Proceedings of the 24th International Conference on Software Engineering* (ICSE), Orlando, Florida, 2002

[6] N.Leveson, Safeware: *System Safety and Computers*, Addison Wesley, 1995.

[7] L.Pullum, *Software Fault Tolerance: Techniques and Implementation*, Artech House, 2001

[8] W.R. Dunn, *Practical Design of Safety-Critical Computer Systems*, Reliability Press, 2002.

[9] Kopetz, H.,*Real-Time Systems, Design Principles for Distributed Embedded Applications*, Kluwer Academic Publishers, 1997.

[10] Conmy, P., Nicholson, M., Purwantoro, Y.,M., and McDermid, J. (2002) *Safety Analysis and Certification of Open Distributed Systems*.

[11] J. A. McDermid, The cost of COTS, *IEE Colloquium - COTS and Safety critical systems* London,1998.

[12] IEC 61508 *Functional Safety of electrical / electronic / programmable electronic safety-related systems* Geneva: International Electrotechnical Commission, 1998.

[13] Tindell, K., "*Analysis of Hard Real-Time communications*", Real-Time Systems,vol 9,pp,147-171,1995.

[14] Jesty, P.H., Hobley, K.M., Evans, R., and Kendall, I., "Safety Analysis of Vehicle-Based Systems," *Proceedings of the 8th Safety-critical Systems Symposium*, 2000.

[15] Raghu Singh. "A Systematic Approach to Software Safety". *Proceedings of Sixth Asia Pacific Software Engineering Conference* (APSEC), Takamatsu, Japan, 1999.

[16] N. G. Leveson "Software Safety: Why, what, and how". *ACM Computing Surveys*, 18(2):125-163, June 1986.

[17] The University of York, *Safety critical systems engineering, system safety engineering,* Modular MSc, diploma, certificate, short courses1999.

[18] The University of York, Heslington, U.K.; www.cs.york.ac.uk/MSc/SCSE.

[19] *The Hazards Forum, Safety-related systems: Guidance for engineers,* The Hazards Forum (1995). London,U.K.;

www.iee.org.uk/PAB/SCS/hazpub.htm.

